The documentation of product configuration systems: A framework and an IT solution

Shafiee, Sara; Hvam, Lars; Haug, Anders; Dam, Michael; Kristjansdottir, Katrin

Published in:
Advanced Engineering Informatics

Link to article, DOI:
10.1016/j.aei.2017.02.004

Publication date:
2017

Document Version
Peer reviewed version

The documentation of product configuration systems: A framework and an IT solution

Abstract. When designing and maintaining a product configuration system (PCS), complete and up-to-date documentation of the system is needed in the form of a product model that outlines the structures, attributes, and constraints of the PCS. Furthermore, up-to-date documentation for the PCS is crucial for maintenance, further development, system quality and communication with domain experts. Product models are the main communication and documentation tools used in PCS projects. Recent studies have shown that up-to-date documentation for the PCS is often lacking due to the significant amount of work required to maintain product models. To address these challenges, this paper proposes an approach for documenting the PCS that is based on the structure, attributes, and constraints modelled within the PCS, in which the product model is generated directly from the PCS. The suggested approach avoids knowledge duplication, as knowledge needs to be maintained within the PCS only. It involves two steps: the first is the building of the initial product model, which is used for the programming of the PCS. In the second step, the product model is generated directly from the PCS and is based on the structure, attributes, and constraints inside the PCS. The product model does not need to be maintained, therefore, outside the PCS. This approach meets the demand for agile documentation and efficient communication with domain experts, and uses the fewest resources possible. Furthermore, to support the framework, an IT documentation system is proposed that is capable of retrieving knowledge from the PCS and thus generating the product model. Our framework and IT documentation system were developed and tested at a case company on five different projects. The results confirm that benefits can be achieved by using the proposed IT documentation system, as time and resources are saved, while the quality of the PCS is improved.

Key words: Product Configuration System (PCS); IT Documentation System; Product Modelling; Agile framework

1. Introduction

Product configuration systems (PCS) are expert systems that support product customisation by defining how predefined entities (physical or non-physical) and their properties (fixed or variable) can be combined [1]. Automation of engineering processes is increasingly prevalent in multiple lifecycle phases such as design, manufacturing and service support [2]. Widely used in various industries, PCS can bring substantial benefits, such as shorter lead times for generating quotations, fewer errors, increased ability to meet customers' requirements regarding product functionality, the use of fewer resources, optimised product designs, less routine work and improved on-time delivery [3,4,5,6,7].

One of the main challenges when using PCS can be a lack of documentation, which can lead to incomplete and outdated systems that are difficult to understand [8]. For a company using a PCS, it is therefore crucial to have an efficient system for documenting the structure, attributes, and constraints modelled within the system, as well as to facilitate communication between PCS developers and domain experts[1]. Documentation is a vital part of all IT projects, as it is used for sharing knowledge between people and reducing knowledge loss, when team members become inaccessible [9]. The documentation of PCS includes modelling, maintaining and updating the product model, and storing all information related to the products' attributes, constraints and rules inside the PCS [1]. Product modelling is a method of representing the structure and knowledge of the product on a relatively visual, abstract level to ensure that they are understandable to all persons concerned.

Four basic representations for modelling product families for PCS are shown in Fig. 1 [10]. The real world is the product knowledge available at a company. A product model presents product knowledge in a structured way, whereas an information model (IT model) is a formal, IT-based representation of a product model, which is usually based on Unified Modelling Language (UML) notation [11]. As an IT model often cannot be understood by most domain experts – especially when it includes complex

[1]Domain experts provide the knowledge for the process in terms of performing the tasks and the data content, in addition to ensuring quality and providing verification support [1].

constraints and rules – the product (or phenomenon) model enables domain experts and configuration engineers to communicate in a common language, thus facilitating future updating and documentation changes [1].

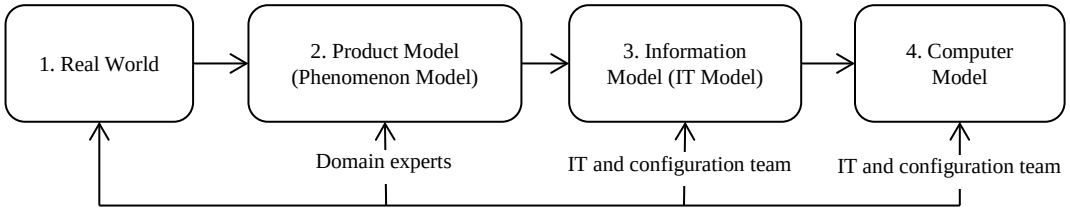


Fig. 1. Four basic representations of product modelling for PCS, revised from Duffy et al. [10].

Previous studies discuss different ways of documenting PCS, which are expressed as product modelling techniques [1, 12, 13, 14, 15]. There are IT tools available to facilitate the documentation process [16]. However, drawbacks to these techniques have been reported as they require manual work to be undertaken, even when IT tools are being used. That is, both the modelling and knowledge modifications have to be maintained separately in the PCS and in the system used for documentation. Accordingly, the knowledge is duplicated, as the knowledge has to be maintained within two separated systems [17]. Thus, maintenance of the PCS requires updates to be made to the PCS (IT model) and the product model, using both time and resources. Moreover, there is a high risk of the documentation of the configuration models being of low quality [8]. Furthermore, a number of projects have highlighted developers' resistance to document product models due to the manual work required and time taken [18].

This paper presents an approach (a framework and an IT documentation system) for generating product models directly from the PCS. To support the framework, we have developed an IT documentation system that is capable of retrieving information from the PCS and representing it using a structure that corresponds with that of a product model. In accordance with the focus of the study, the following research questions have been developed:

1) How can we document PCS using agile documentation methods?
2) How can an IT system be developed to support agile documentation for PCS?

To answer the research questions, we explored the literature in order to identify ways to construct such a framework and to determine the requirements of the IT documentation system that would support the framework. Both the framework and the IT documentation tool were tested at a case company on five projects in total. The tests indicate that an IT documentation system not only significantly decreases the resources and time needed for documentation, but also improves the quality of the PCS.

This paper is organised as follows: part 2 elaborates on the methodology for the research, explaining the two key phases. Part 3 discusses the relevant literature, while part 4 details the framework and the development of the IT documentation system. Part 5 presents the results of the case study and part 6 discusses the results, examining the usability of the applied framework and the IT documentation system. Finally, part 7 answers the research questions and discusses the limitations and conclusions of the research.

2. Research method

To achieve the overall goal of improving the process for PCS documentation using agile methods, this research is structured into two phases. The first phase focuses on the development of the framework and the IT documentation system, while the second phase tests the suggested framework and the IT documentation system.

2.1. Development of the framework and the IT documentation system

In order to develop our framework, first, we studied the available literature on documentation methods, agile documentation methods and product modelling. To provide a foundation for the proposed framework, we studied why and how agile documentation techniques are of benefit. Consequently, the aim to propose a structured documentation process that automatically generates the required documentation in the form of a structured product model was established. This rationale for choosing such a process was to eliminate the duplication of knowledge and to facilitate the maintenance of PCS from one place only. Next, the literature on product modelling techniques was evaluated and the most suitable method was selected for this project in accordance with the level of visualisation required.

For the development of the IT documentation system, we examined the literature, investigating the methodologies and requirements that have been used, maintaining a special focus on both IT and PCS-based projects. Using the proposed framework, updating and maintaining occurs within the permanent model only. Accordingly, we decided to introduce the product structure as a product model within the PCS, with a view to investigating the possibility of generating documentation via the structure used inside a commercial PCS. Based on evidence from the literature [19] and our experience of working with multiple commercial PCS, we recognised that most of commercial PCS contain the knowledge required to generate a product model that could be used for documentation. After further investigation, we developed an IT solution for documenting the knowledge within a PCS, which could be developed in a short time and without significant investment.

Fig. 2 illustrates the complete process of developing the framework and the IT documentation system, and explains the research steps and methods used for the different phases.

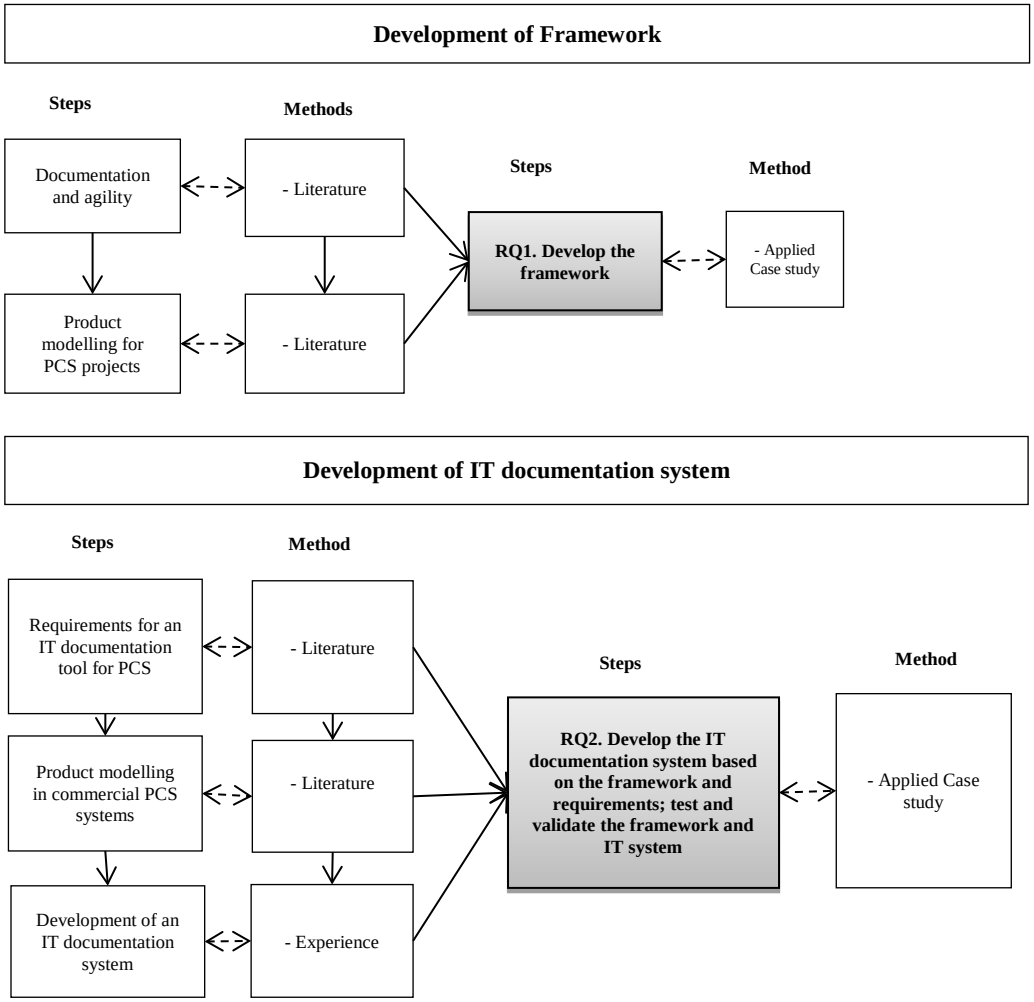


Fig. 2. Methodology for developing the framework and the IT-based documentation system.

2.2. Evaluating the usability of the system

We tested the proposed framework and the IT documentation system in order to discover: (1) whether the structure, attributes and constraints of a commercial PCS could be extracted, (2) how to extract knowledge from a PCS and (3) whether the generated documentation would be sufficient for both the domain experts and configuration experts at a company.

The requirements for developing the IT documentation system are listed later in the paper (section 3.4). Usability is 'the extent to which the IT system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use' [20]. Nielsen [21] lists five characteristics of usability, as follows: (1) learnability (how fast a new user can begin to work with a system), (2) efficiency (level of productivity), (3) memorability (i.e. amount of relearning required after spending time away from the system), (4) low error rate and (5) satisfaction. In our study, memorability was not relevant, as approximately two years have passed since the system at the case company was launched and it has been used continuously. Based on the definition given of usability we formulated the following interview questions:

1. How much time is required to learn the IT documentation system?
2. How much time (resource time) is saved by using the IT documentation system, compared with older methods, such as Excel spreadsheets?
3. How much error reduction occurs in PCS with the use of the IT documentation system?
4. What is the users' level of satisfaction and acceptance of the IT documentation system?

In relation to the question 2, it should be noted that by time we mean resource time that includes time of the configuration team, domain experts and steering committee. Based on this question, we assess the saved resource time as the result of using the IT documentation system. The saved time from resources also impacts the total lead-time when developing and maintaining PCS and is therefore of a high importance.

The data was collected by interviewing different stakeholders at the company in order to reflect different expectations with regard to the IT documentation system. The main stakeholders included domain experts, members from the configuration team and the steering committee. The results provided a list of further possible improvements for the IT documentation system, as well as indicating the end users' levels of satisfaction and acceptance of the system.

3. Literature review

The literature review aims to find theories for modelling and documenting PCS (sections 3.1 and 3.2), and for developing the IT documentation system in general (sections 3.3 and 3.4). Section 3.1 and 3.2 focus on the modelling and documentation of PCS, and on agile documentation principles. Section 3.3 elaborated on product modelling in commercial PCS and Section 3.4 discusses the requirement for documenting IT projects in general and PCS specifically.

3.1. Product modelling for PCS projects

This section reviews the various product modelling techniques suggested in the literature for use in PCS projects. As mentioned in part 1, modelling tools are used for the communication and documentation of PCS. Product modelling is used to handle the growing complexities of software development, enabling engineers to work and communicate at higher levels of abstraction [22]. In addition, these techniques increase the ability to share knowledge between units, which can contribute significantly to an organization's performance [23]. Different modelling techniques are available for PCS [13, 24, 25, 26, 27, 14, 28, 15]. UML is a general-purpose visual modelling language designed for the specification, visualisation, construction and documentation of the artefacts of a software system [29]. It encourages designers to formalise their implicit knowledge and makes knowledge extraction easier. Felfernig et al.

[30] describe a method for using UML as a configuration development tool to support graphical notation. A detailed analysis of the product range is necessary – the use of the Product Variant Master (PVM) together with CRC cards is suggested for this purpose (Fig. 3), both being based on UML techniques. The PVM presented by Hvam [31] displays product knowledge in a structured format, focusing on three different aspects: the customer's view, the engineering view and the production/part view. The use of CRC cards was first put forward by Beck and Cunningham [32] as a way to teach object-oriented thinking. Hvam et al. [33] have since proposed several revisions for CRC-card use in PCS projects, in which the classes utilised are described in more detail.

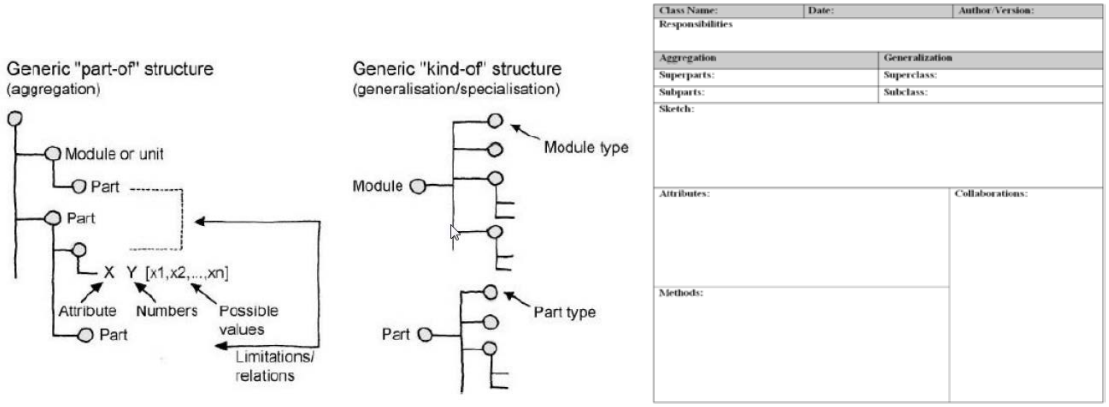


Fig. 3. Structure of the PVM and CRC cards [1].

Renzl [34] emphasises that documentation systems are critical in the sharing of knowledge. Rask [35] addresses documentation and maintenance requirements by emphasising the importance of traceability, verification and versioning. Application of the Dependency Structure Matrix (DSM) method could provide natural and transparent documentation of the 'knowledge base' [36]. Multiple IT tools have been developed to assist with documentation in IT and PCS projects. For example, the Computer Aided Software Engineering (CASE) tool applies a set of tools and methods, generating a software system to support the desired end result: i.e. a defect-free and maintainable software product [37,38]. The CASE tool can be used for software documentation; however, CASE tools are not specifically designed for PCS, but rather for the general development of IT projects. Elgh [39] proposes a framework consisting of an information model and underlying principles to be used when developing a design automation system for quotation preparation. Existing research on PCS documentation indicates that additional specifications were introduced in 2007 for a PCS IT documentation system called Product Model Manager – an environment which provides a simple overview of the product elements in the form of CRC cards and PVMs [12]. According to the principles of model-based systems engineering (MBSE), it can replace traditional documentation methods and facilitate the collaborations by representing all aspect of the systems engineering [40, 41, 42]. System Engineering Applications (SysML) reuses a subset of UML introduced as a standardized general purpose graphical modelling language for capturing complex system descriptions in terms of their structure, behaviour, properties, and requirements [43]. Our suggested method for IT documentation system and using UML as the modelling language can be considered a special case of MBSE as the suggested system capture the knowledge in the PCS and create domain (phenomenon) models. A functionality analysis for Product Model Manager was compared with types of software commonly applied for documenting PCS [16], which showed that the new software offered successful PCS documentation, although the common challenge of additional manual work being required remained. As such, there is still a critical need in the industry for an efficient automatic solution to be found for documenting the knowledge of PCS, communicating with domain experts and validating product knowledge.

3.2. Documentation and agility

This section discusses the agile documentation lifecycle, based on UML principles. Communication between IT personnel (software developers and modellers) and domain experts is important for software development and is a success factor when discussing new changes in software development projects and teams [44]. Cockburn [45] claims that clear communication is necessary for inexpensive and fast project implementation. Selic [46] explains agile documentation by elaborating different steps for design and development. Even in the earliest stages, documentation must be adequate to serve its primary communications purpose. Agility and avoidance of duplicate knowledge are essential in software documentation [36, 46]. The UML state chart shown in Fig. 4 represents the lifecycle of an agile model [47]. It is possible to create a temporary model, which can be used to communicate initial ideas but is not made permanent until the model is completely clear to everyone. After the permanent model has been developed, all maintenance tasks are performed within that model; the temporary model is skipped. This approach creates value and makes programmers more willing to document their work. The next step determines the content and the level of detail required for efficient software maintenance [48]. Agile methods promote lean, light and easy documentation.

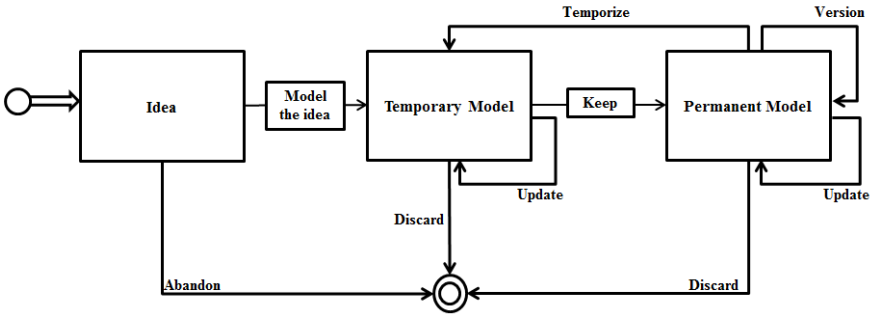


Fig. 4. A UML state chart that depicts the lifecycle of an agile model. The idea is modelled temporarily to aid communication and modelling in the initial stages. After the validation of the temporary model, the permanent model is generated [47].

3.3. Product modelling in commercial configuration systems

This section discusses the available literature regarding commercial PCS, their structures and levels of maturity in order to demonstrate the capabilities of commercial PCS to perform as documentation systems.

The literature discusses many commercial configurators that are currently available, such as Tacton, Encoway and SAP [49]. There are benefits, strengths and weaknesses to each of the available configurators. Friedrich et al. [49] describe modern PCS as systems that must provide a mechanism to abstract the underlying technical representations inside the PCS as much as possible during the modelling phase. The typical elements found in such tools include compatibility tables (which non-IT specialists can understand), a user-oriented rule language (including appropriate editing support), a graphical representation of bills-of-material structures, and testing and tracing support [49].

Several researchers put forth suggestions for better structures for commercial PCS, listing the requirements. For example, Tiihonen et al. [8] evaluate the modelling efficiency and performance of PCS in the views of several domain experts and present a list of requirements for a future, web-based commercial PCS. Myllärniemi et al. [50] present a feature configuration model in which the attributes, composition structure and constraints of certain elements, features and properties are given, with an explanation of how these aspects can be combined to create a valid product.

A number of researchers (as mentioned in previous paragraphs) agree that available commercial PCS should be capable of communicating with non-IT specialists through visualised modelling structures, but modern PCS remain immature in this regard. The existing literature indicates that modern PCS does not include sufficient documentation systems, but they do have the capability to generate the input for a documentation system automatically.

3.4. Requirements for an IT documentation tool for PCS

In Table 1, we identify and prioritise the most important requirements for IT documentation tools for PCS based on the literature studied. The requirements are labelled as either general requirements for documenting IT systems or specific requirements for documenting PCS. The requirements are sorted according to importance, from the most to the least important.

Table 1. Requirements for PCS documentation

	Requirements	**Description**	**General/ Specific**
1	Accessibility and ease of use	Providing accessibility to end users in a format that they can understand. In addition, it is important that documentation can be easily generated [39, 51].	General (IT)
2	Model history overview	Management of the changes made to the model and the ability to revert to a previous version [51].	Specific (PCS)
3	Model tree structure	New primary view with an unchangeable tree structure for maximum clarity [16].	Specific (PCS)
4	Navigation ability	An intuitive user interface displaying the model documentation and allowing the user to navigate and search for knowledge in the documentation system [1].	General (IT)
5	Change requests and notifications	Allowing users to comment on documentation and make requests for changes by sending change requests to the modeller responsible [47, 51].	General (IT)
6	Updated	Allowing the easy and fast updating process by the modellers [52].	General (IT)
7	Entering changes and updating in one place	Avoiding errors when updating the PCS and avoiding knowledge duplication. The documentation system should receive changes automatically [47].	General (IT)
8	Version comparison	Allowing historical comparisons between different versions of documentation [35].	General (IT)
9	Broad network access	The user should have access to online software, irrespective of geographical location (important for regional offices) [47].	General (IT)
10	Cost efficiency	The system should save the company resources, compared to existing systems [52].	General (IT)
11	Language	For the widest possible usage, the language of the tool should be in English (perhaps with multilingual support) [51].	General (IT)
12	Access management	The system should have a database of user groups, allowing for different rights of access for model editing [53].	General (IT)
13	Hyperlinks	Allowing the user to create links to external references, such as drawings and documentation [51].	Specific (PCS)
14	Multiple views at the same time	Allowing multiple views at the same time on multiple personal computers [1].	General (IT)
15	Active search	A visualisation of the search function [1].	Specific (PCS)
16	Flexible structure	The system should be adoptable for integration with, or extension of, other systems [1, 51].	Specific (PCS)
17	Integration	Integration with other systems in the company, e.g. Enterprise Resource Planning (ERP) systems and other data bases, should be possible. The person responsible should be informed by email each time a change is made to the document (whether version or name) [51].	Specific (PCS)

4. The proposed approach for the agile documentation of PCS

This chapter discusses: (1) the framework development, (2) the structures within a PCS for the retrieval data and the setup of a product model, and (3) the IT tool, in terms of technical details.

4.1. An agile documentation framework for PCS

This section presents the proposed framework for the agile documentation of PCS. The main idea of this framework is based on the agile lifecycle model [47], as described in Section 3.2. The focus is upon the creation of a temporary model version, which is made permanent once it has been proved to be viable and trustworthy as a documentation system.

The method selected for structuring the product model, which is also used for the documentation, utilises the Product Variant Master (PVM) together with Class Responsibility Collaboration (CRC) cards We have chosen the PVM and CRC cards as methods for structuring the product model as they are described well in the literature [1] and enable a high level of visualisation, even where complex products are concerned.

First, the initial product model is built using specific modelling techniques (in this study, via the PVM and CRC cards). This phase is the temporary documentation phase. Based on the initial modelling, the model is programmed into the PCS, and models similar to the PVM/CRC-card approach are then generated from the PCS directly via the documentation system. For future versions of the PCS, updates will be made from within the PCS and the updated product models will be generated directly from the PCS. Fig. 5 demonstrates the processes for the suggested agile documentation of the PCS.

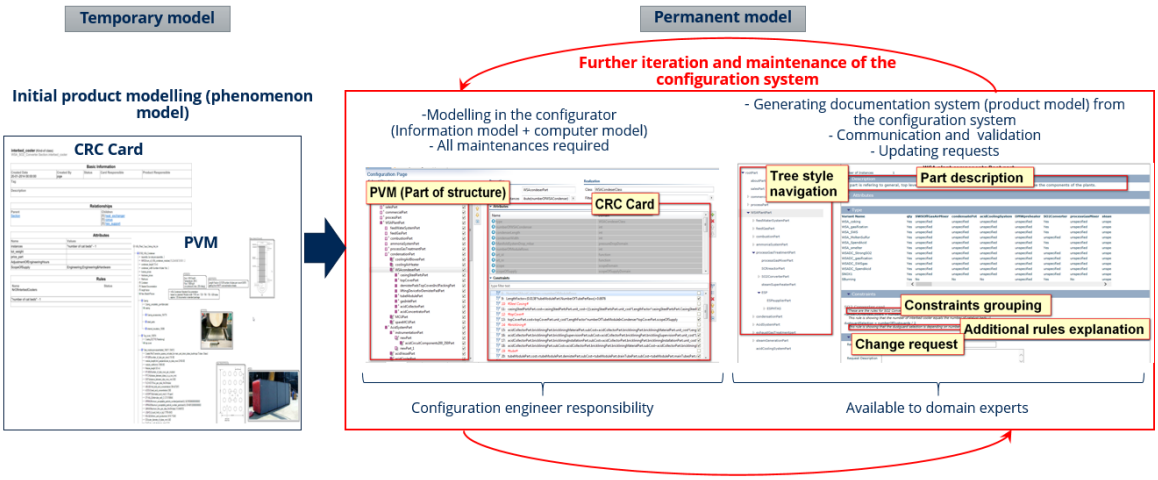


Fig. 5. The process of generating documentation from the PCS (the photographs will be shown in the case study).

The product model is generated based on the structure inside the PCS; the configuration engineer simply has to upload his PCS model to the documentation system. With this approach, the IT experts will provide the configuration engineers with the ability to control their models, such as showing/hiding different parts or providing users with descriptions. The model is depicted as a tree structure inside the documentation system. The documentation system is capable of controlling the different versions of models by storing them in the history. Domain experts can access the system, depending on their rights, which are managed by the system administrator. The experts can see the tree structure, navigate between variants and constraints, and ask for changes to be made to any part of the model by sending through change requests. When a request is sent, an email will arrive to the configuration engineer responsible for the model, who will be able to see which part of the model the change request refers to specifically. Once changes have been made to the model and it has been tested, the model will be uploaded to the documentation system as the latest version. Domain experts are not able to make changes to the documentation system because the goal is to keep all the changes within the PCS to avoid errors.

Moreover, the licenses for PCS modelling are limited, domain experts are not capable of understanding the IT language used inside PCS without additional training, and a specific person should be responsible for updating and maintaining the system.

4.2. Development of the IT documentation system

This section describes the process of developing the IT documentation system, starting with an explanation of the correlation between the structures in commercial PCS and the structure of the product model, which is followed by a description of how the knowledge is exported from a commercial PCS to the documentation system.

4.2.1 The corresponding structures in PCS and product models

For this study, commercial PCS available on the market were evaluated. As shown in Fig. 6, the model structures within these commercial PCS correspond to the PVM/CRC-card approach. More specifically, all of the components and sub-components – including all of the attributes, constraints and rules – form a tree structure. The goal is to transfer and translate all the relevant knowledge to the documentation system and thus to generate PVMs and CRC cards automatically.

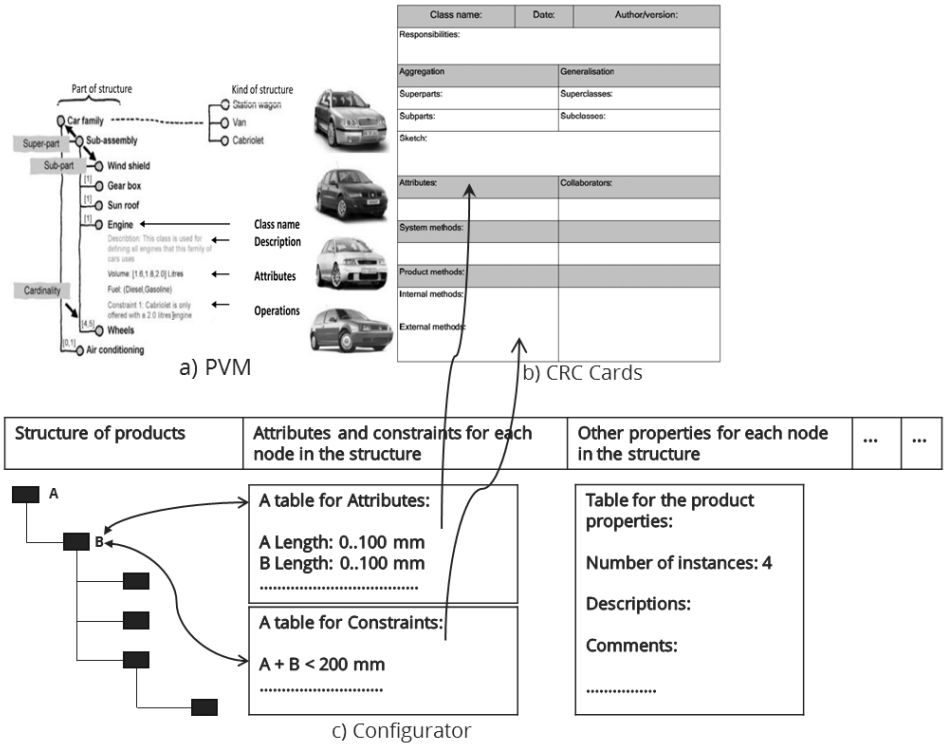


Fig. 6. Relations between the product model and the common structure in commercial PCS. PVM and CRC-card structure (a & b) [1]; traditional structure of commercial PCS (c).

The tree structure represented in the PVM describes the structure and the available variants shown. On top of the attributes and constraints listed in the PVM structure, the CRC cards are used to describe individual classes in more detail [8], as shown in Fig. 6.

Common drawbacks of product knowledge models in commercial PCS modelling environments are the lack of additional explanation regarding rules/variants, the unavailability of product figures and the lack of a product hierarchy [19]. Furthermore, most of the knowledge is stored in IT language. These drawbacks indicate that commercial PCS seem to fall short in terms of the functionality required for documenting product models.

4.2.2 Setting up the IT documentation system

This section explains the development of the IT documentation system at the case company. The platform for the IT documentation system is a content management system (in this case, Microsoft SharePoint). The original documentation format generated from a commercial PCS model (Tacton was used here) is based on the xml format; by renaming the model file extension as .xml, the documentation can be read/ parsed.

With this system, the PCS models will be maintained in a document listed in the content management platform called the 'Model Workspace', from which the IT documentation system will read the files. The Model Workspace is based on a fairly standard SharePoint Team site, which places the models under version control (minor/major versions) and has check-out/check-in functionality to lock the models while updating.

To render the model documentation with the required navigation, layout and content syntax, a transformation/interpretation of the native PCS xml is needed. This process parses and transforms the model structure into JSON objects, which can be rendered using feature-rich JQuery components. Fig. 7 shows the interactions between SharePoint and the UI Renderer, which, together, make up the IT documentation system.

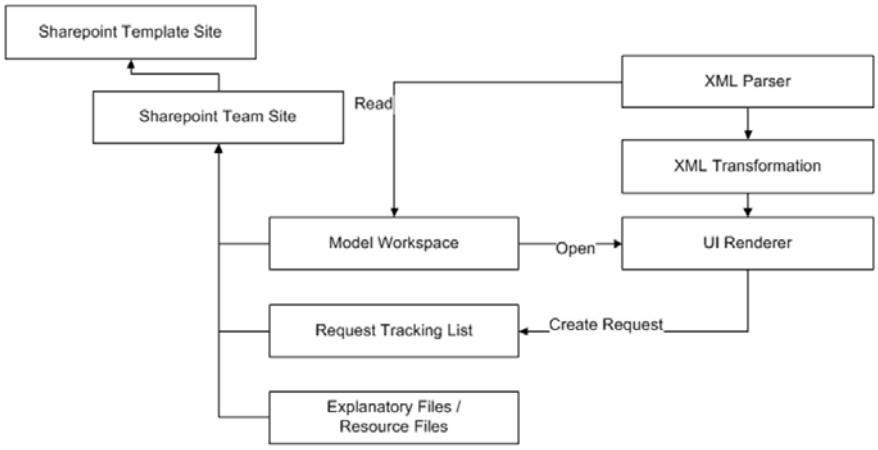


Fig. 7. Interactions between the Content Management System and the UI Renderer (IT documentation system).

For integration purposes, we used the SharePoint JavaScript Object Model (JSOM). This JavaScript API is used for receiving lists and list items from SharePoint, as well as creating requests back in SharePoint. The following is an example of how a request is registered to the request tracking list:

```
var clientContext = new SP.ClientContext('/sites/20');
var oList = clientContext.get_web().get_lists().getById('{7A6EFEDF-5D65-4241-A6E0-E84CF2F3312D}');
var itemCreateInfo = new SP.ListItemCreationInformation();
var oListItem = oList.addItem(itemCreateInfo);
oListItem.set_item('Title', $('#component_name').text());
oListItem.set_item('Body', $('#rfctext').text());
oListItem.set_item('Model', 'testtest2.dk');
oListItem.update();
clientContext.load(oListItem);
clientContext.executeQueryAsync(function() {
  alert('Item created');
  $('#rfctext').text('')
}, function(sender, args) {
  alert('Request failed. ' + args.get_message() + '\n' + args.get_stackTrace());
});
```

Below, the proposed approach for developing an IT documentation system is explained further and is tested via a case study.

5. Case study

The proposed approach (the framework and the IT documentation system) was tested on five PCS projects. The case company, which operates globally, specialises in catalyst production and process-plant technology. A project team was formed at the case company, including two full-time researchers from the university, together with a software developer and a configuration engineer from the company, who spent approximately half of their time working on the project over three months. During the 10 months before the development phase commenced, the architecture of available documentation systems and the framework for the future documentation system were discussed and outlined. The major role of the researchers was to develop the proposed approach, extract and analyse knowledge, prioritise the requirements of the documentation system, and facilitate interviews to evaluate the models generated by the system. A software developer programmed the IT documentation system to extract the structure, constraints and attributes of the commercial PCS used at the company from the PCS itself. The total cost of the documentation system is calculated to be 6000 USD, which includes requirement analysis, making the architecture of the system, development, debugging and training. It is estimated that the documentation system had positive return on investment in the first year due to saved time and resources on different stages of the project.

The case company has five operational PCS, which support the sales and engineering processes. All these five system are similar as they are configuring the similar highly engineered complex products or processes. This enabled us to test the documentation system within five different environments at the same company, with varying levels of complexity. The IT documentation system was tested during the development and operation of all five projects. The characteristics of the different projects are provided in Table 2.

In each of the projects, training workshops were held to demonstrate the IT documentation system and to obtain feedback. Each workshop lasted between 30 minutes and 1 hour. The content of the workshops included the introduction of the IT documentation system, training on the use of the system and open-forum segment that encouraged feedback and improvement suggestions. All the examples given in this section relate to Project 2.

Table 2. Background information for the projects used in the case studies

Projects	Project 1	Project 2	Project 3	Project 4	Project 5
Time frame for development, running and maintenance (months)	6	24	12	6	4
Complexity of the project (number of attributes and constraints in the PCS)	Medium	Great	Great/Medium	Medium	Medium
No. of employees involved	4	10	6	4	3
No. of workshops	3	6	4	2	3
No. of feedback meetings	4	15	4	2	3

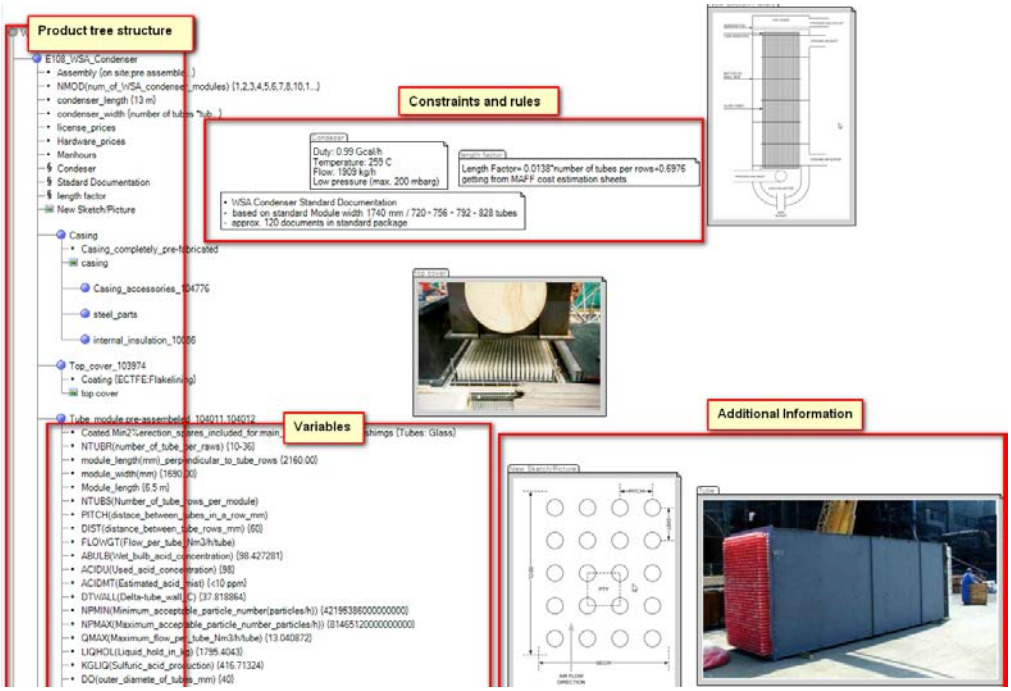
To measure the complexity of PCS, Brown et al. [54] categorize them into three major components; 1) execution complexity, 2) parameter complexity, and 3) memory complexity. Execution complexity covers the complexity involved in performing the configuration actions that make up the configuration procedure and the memory complexity refers to the number of parameters that system manager must remember. In this paper, the parameter complexity is the most important, as it measures the complexity involved in providing configuration data to the computer system during a configuration procedure [54]. Therefore, we assessed the parameter complexity in terms of two major parameters inside the PCS: attributes and constraints (Table 3).

Table 3. Complexity assessment in terms of parameters in PCS

	No. attributes	**No. constraints**
Small complexity	500 - 1300	200-800
Medium complexity	1300-2000	800-1200
Large complexity	>2000	>1200

5.1. Phase 1: Initial product model

In the initial phase, the product assortment, the individual components and the relation between these components were modelled using the PVM and CRC cards. The PVM and CRC cards are used only when the PCS system is being developed; after the system has been developed, the models are generated from the PCS and maintained within the documentation system. The initial PVM was used for communicating with domain experts in the early phases of the project. Fig. 8 shows part of the PVM used initially to model the product assortment. Detailed information regarding different nodes in the PVM are described on the CRC cards.



Class Name: Baffle plates		
Responsibilities: The class contains knowledge about the baffle plates' arrangement, dimensions and capacities.		
Aggregation	**Generalisation**	
Super parts: None	**Superclass:** None	
Subparts: None	**Subclass:** None	
Sketch: 		
Attributes: Length (8 mm) Width (1900.00 mm) Delta tube wall (45.6352 C) Maximum flow per tube (54.584 Nm^3/h/tube) Minimum acceptable particle number (54662888410000000 particles/h) Distance between baffle rows (90 mm)		**Collaborations:** Drain tubes, Side tubes
Methods: Module width calculation: number of plates = 1, then 8-9 modules number of plates = 2, then 18-25 modules		

Fig. 8. The initial PVM and CRC-card structure of the product (during PCS development).

5.2. Phase 2: PVM and CRC-card generation from the PCS

The PCS model contains the knowledge required for the documentation to be generated, including the tree structure of the product, along with all the variants and rules. Fig. 9 shows the structure and knowledge within a commercial PCS (in this case, Tacton) as a PVM (tree structure) and a CRC card.

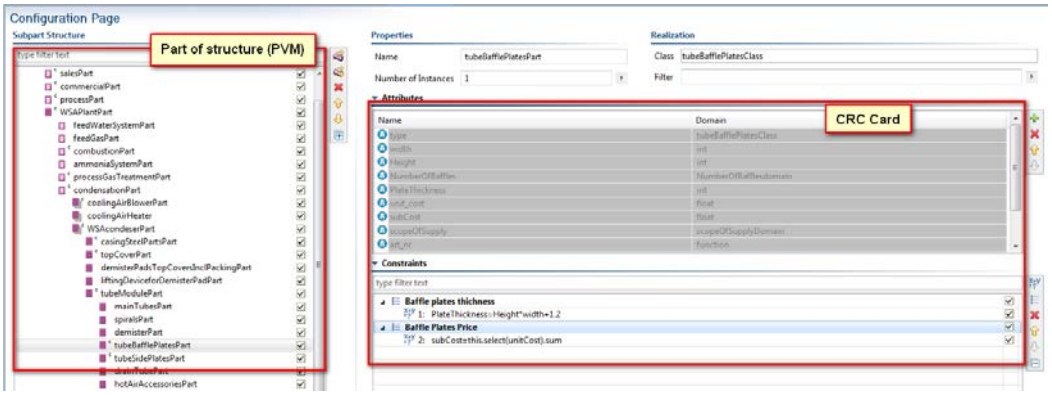


Fig. 9. Available PVM and CRC-card framework in the commercial PCS at the case company.

It should be noted that, even if all the knowledge was accessible to the domain experts, they would not be able to understand the structure, attributes and rules inside the PCS. Therefore, to make both the logic inside the PCS and the reasoning behind the different calculations and restrictions visible/understandable, an IT documentation system can be used. The IT documentation system covers the aspects that appear in the requirement list (Table 1), such as the attributes of the product model (tree structure, components, attributes, features, constraints and name domains), and gives a tree-style navigation of the model's main structure. It is possible to show or hide parts, attributes, rules and other sections within the model – collapsible panels appear in the content sections of the display. The user is able to submit requests/comments to the request tracking list directly from the documentation view, and the request's context is included automatically in the request tracking information. Additionally, rich syntax is available to enrich the documentation and make it more readable to business users and experts. Tags can be used to add links or text, or to hide/show knowledge. Fig. 10 illustrates the tree structure of the PVM and the CRC cards, as generated by the IT documentation system based on the PCS model.

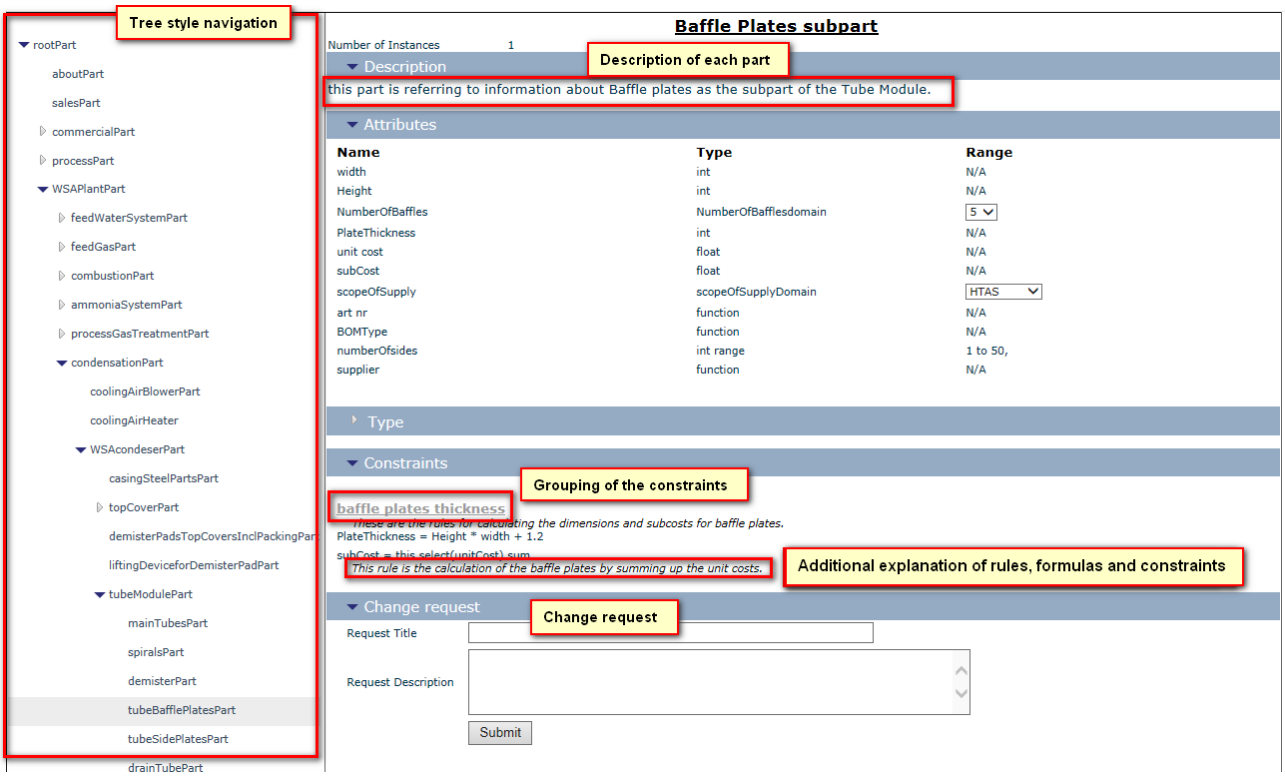


Fig. 10. Knowledge generation in the PCS and PVM/CRC structure inside the IT documentation system.

A comparison of the structure in Fig. 9 (PCS structure) with that in Fig. 10 (IT documentation system) reveals the differences between the two ways of documenting. The IT documentation system emphasises the descriptions and translations for each part, as well as the tree structure and easy navigation of the model. The 'change request' part includes requests from domain experts for changes that need to be made to the model. To make the tool complete, a 'change request' field with navigation possibilities is necessary. Every change request must include the time, the part, the name of the user, the title and the request made, as shown in Fig. 11. The system also has the ability to display pictures, as with PVMs and CRC cards, which was not necessary in the example of the case company.

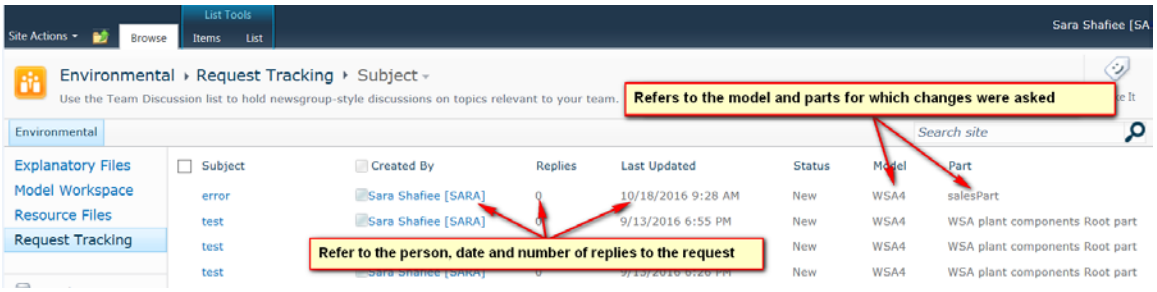


Fig. 11. Change request overview and history.

6. Discussion

Employees at the case company were interviewed to assess the IT documentation system. The interviewees were selected from among the configuration team, domain experts and members of the projects steering committee. The main results from the interviews are summarized in Table 4.

Table 4. Usability interviews

Questions	Configuration team	Domain experts	Steering committee and top managers (3 persons in total)
How much time is required to learn the IT documentation system?	0.5–1 hour	2–3 hours	1–3 hours
How much time is saved using the IT documentation system, compared with older methods, such as Excel spread sheets?	50%–60% of the total time	70%–80%, compared with updating Excel spreadsheets and attending meetings	30–40%
To what extent are errors reduced in PCS due to the use of the IT documentation system?	30%–40%	20%	20%
What are the users' levels of satisfaction with, and acceptance of the IT documentation system?	Very high	Very high	High

The results indicate that the interviewees found the learning processes and use the IT documentation system to be considerably easier than those for previous communication/updating systems, such as Excel spreadsheets, Visio files, CAD files and ERP systems. The differences in the answers obtained depended on the interviewees' familiarity with the model, and with PVMs/CRC cards. The findings regarding time saving vary in ranges, depending on the product complexity. The IT documentation system prevented the overwriting of knowledge and reduced errors in PCS because the domain experts had a higher level of understanding of the system, compared with their understanding of previously used systems. Furthermore, the results from the interviews indicate that 1) resource time is saved because of the efficiency of the new system and updating the knowledge in only PCS, 2) the quality of PCS is improved because of the good communication and verification of knowledge inside PCS, and 3) acceptance/satisfaction levels are high

due to a nice user interface and access to the knowledge for testing and debugging. This system provides users with documentation, validation and communication for domain experts and the configuration team.

We analyse the IT documentation system to evaluate to what extent it is providing the requirements discussed in Section 3.4 using compliance matrix. In order to show the compliance of developed IT documentation system to overview of applicable requirements for documenting PCS listed in Table 1, a compliance matrix is provided (Table 5). This matrix formalizing requirements verification with respect to the developed system; which directly uses the dimensions of Table 1.

Table 5. Compliance matrix for requirements verification with respect to Table 1 and developed IT documentation system

No.	Configuration Documentation System Compliance Matrix				
	Requirement	Compliance			Comments
		Full	Partial	None	
1	Accessibility and ease of use	X			Easy fast generation of product documentations. The system should have ease of accessibility both for configuration team and domain experts.
2	Model history overview	X			The system controls all the uploaded versions and saves them with additional details.
3	Model tree structure	X			Tree structure view is available that represent the hierarchy of the actual product structure.
4	Navigation ability	X			A powerful navigation ability even for different attributes in the constraint part.
5	Change requests and notifications	X			A very nice user interface and easy way both to generate and respond to change request is provided.
6	Updates	X			Updates take place in the PCS the PCS model is then uploaded to the IT documentation system in easy manners.
7	Entering changes and updating in one place	X			As changes and updates are all done in the PCS model. The knowledge is maintained in one place which eliminates duplications.
8	Version comparison	X			Historical comparisons between different versions is available.
9	Broad network access		X		The IT documentation system is implemented on the case company's internal network, where there are security requirements to get connected to the system.
10	Cost efficiency	X			Calculated Return On Investment (ROI) was positive in the first year after the IT documentation system was launched.
11	Language			X	The system is available in English. Further development will include other langue for regional offices.
12	Access management	X			The platform provides different access rights for different group of users (e.g. configuration team and domain experts).
13	Hyperlinks			X	It will be included in the next version of the system especially for accessing drawings of the components.
14	Multiple views at the same time	X			There is no constraints on how many users can access the system at the same time.
15	Active search		X		The search is possible but not in a very strong and visualized way.
16	Flexible structure	X			The IT documentation system is adoptable to integrations, extension and etc.
17	Integration			X	The system integrated with other documentation systems in the company especially the one which is developed based on SharePoint.

7. Conclusion

As product models become more complex, documenting a product model becomes more time consuming [55]. One of the reasons for this is that the updating of the product model must be completed manually. The importance of proper documentation for a PCS is related to the vital knowledge modelled inside the system, which needs to be changed and updated frequently. Besides this, there is a need for knowledge validation by domain experts in order for a reliable system to be achieved. Following a review of the available literature, this paper proposes an agile documentation framework for PCS.

We suggest a framework based on the agile model and previous literature, through which a temporary model can be created. Subsequently, the PVM and CRC cards are generated from the configuration system, based on the structures, attributes and constraints within the PCS. The proposed documentation framework saves time by removing unnecessary manual tasks, such as recording product knowledge multiple times and updating tasks, from the process. This systematic documentation approach gives domain experts ownership of the processes of PCS development and maintenance. The system also facilitates communication between an IT group and domain experts, which has been reported to be one of the main challenges in configuration projects [9]. Additionally, the proposed approach turns the PCS into a comprehensive documentation tool that can be used as a product-knowledge database and in which all changes can be made in one place: within the PCS.

This practical approach has been developed as an IT documentation system that supports the framework's principles. The proposed IT documentation system was implemented using the xml (standard) and JavaScript format at the case company. With this system, the whole configuration team benefited from agile documentation, uploading their PCS models. Thus, domain experts were empowered to check the validity of the PCS continuously. Technical development of the system was quick and easy; the only challenge observed relates to the upgrading of the commercial PCS, as the IT documentation system has to be tested and debugged after each new version is installed (approximately once a year).

Structured interviews, conducted in the testing phase to measure the usability of the new IT documentation system, indicated that the proposed method was useful in facilitating the understanding and debugging of the system. However, the approach might not be applicable to all kinds of commercial PCS (due to the differing structures of PCS). Various challenges might arise, depending on the mind-sets and cultures at different companies. We suggest, therefore, that future research should test the system using different projects, companies, types of software and platforms.

As developing an IT system based on a framework requires not only the availability of the company, but also considerable time and resources, we tested the approach at one company only, which limits the paper's generalisability. Focusing one case company, however, allowed us to gain an in-depth understanding of how the framework operates, as well as providing a detailed loop of improvements and simplifications required in the development of the system. The case company was chosen because it is an engineering company that produces highly engineered, complex products, and currently utilises PCS to support sales and engineering processes. The firm reported difficulties with the maintenance of their PCS, as well as with a lack of documentation for use in communications with domain experts. We assume that if the proposed framework can solve such challenges at one company, other companies with the same level of complexity or with less complex products might also benefit from use of the framework.

8. Acknowledgement

This research was conducted at the company Haldor Topsoe A/S. Permission to undertake the project and the financial support are gratefully acknowledged.

References

[1] L. Hvam, N. H. Mortensen, J. Riis, Product Customization, Springer Verlag, Berlin, 2008.
[2] W. J. Verhagen, B. de Vrught, J. Schut, R. Curran, A method for identification of automation potential through modelling of engineering processes and quantification of information waste. Advanced Engineering Informatics, 29(3) (2015) 307-321.
[3] V. E. Barker, D. E. O'Connor, J. Bachant, E. Soloway, Expert systems for configuration at digital: XCON and beyond. Communications of the ACM, 32(3) (1989) 298-318.
[4] L. Hvam, S. Pape, M. K. Nielsen, Improving the quotation process with product configuration. Computers in Industry, 57(7) (2006) 607-621.
[5] T. Petersen, Product configuration in ETO companies, in: T. Petersen, Mass customization information systems in business. In T. Blecker (Ed.), Igi Global, 2007, pp. 59-76.
[6] L. Ardissono, A. Felfernig, G. Friedrich, A. Goy, D. Jannach, G. Petrone, R. Schafer, M. Zanker, A framework for the development of personalized, distributed web-based configuration systems. Ai Magazine, 24(3) (2003) 93.
[7] C. Forza, F. Salvador, Product Information Management for Mass Customization, Palgrave Macmillan, New York, 2007.
[8] J. Tiihonen, M. Heiskala, A. Anderson, T. Soininen, WeCoTin – A practical logic-based sales configurator. AI Communications, 26(1) (2013) 99-131.
[9] F. Paetsch, A. Eberlein, F. Maurer, Requirements engineering and agile software development, in: IEEE 12st International Workshop on Enabling Technologies, Infrastructure for Collaborative Enterprises, 2003, pp. 308.
[10] A. Duffy, M. Andreasen, Enhancing the evolution of design science, in: Proceedings of International Conference on Engineering Design (ICED), 1995, pp. 29-35).
[11] P. Kruchten, The Rational Unified Process: An Introduction, New York, Addison-Wesley, 1998.
[12] A. Haug, L. Hvam, The modelling techniques of a documentation system that supports the development and maintenance of product configuration systems. International Journal of Mass Customisation, 2(1/2) (2007) 1-18.
[13] A. Haug, L. Hvam, Representation of Industrial Knowledge as a Basis for Developing and Maintaining Product Configurators. Technical University of Denmark, Department of Management Engineering, Operations Management, 2008.
[14] Z. Jinsong, W. Qifu, W. Li, Z. Yifang, Configuration-oriented product modelling and knowledge management for made-to-order manufacturing enterprises. The International Journal of Advanced Manufacturing Technology, 25(1-2) (2005) 41-52.
[15] D. Yang, R. Miao, H. Wu, Y. Zhou, Product configuration knowledge modeling using ontology web language, Expert Systems with Applications, 36(3) (2009) 4399-4411.
[16] A. Haug, A software system to support the development and maintenance of complex product configurators, The International Journal of Advanced Manufacturing Technology, 49(1-4) (2010) 393-406.
[17] S. Shafiee, L. Hvam, K. Kristjansdottir, An agile documentation system for highly engineered, complex product configuration systems, in: 22nd EurOMA Conference: Operations Management in an Innovation Economy. Neuchatel, Switzerland, 2015.
[18] J. D. Herbsleb, D. Moitra, Global software development, IEEE Software, 18(2) (2001) 16-20.
[19] G. Friedrich, D. Jannach, M. Stumptner, M. Zanker, Knowledge engineering for configuration systems, in: A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration From Research to Business Cases, Morgan Kaufman, 2014, pp. 139-155.
[20] W. ISO, Ergonomic Requirements for Office Work with Visual Display Terminals. Geneva: The International Organization for Standardization, 45, 1998.
[21] J. Nielsen, The usability engineering life cycle. Computer, 25(3) (1992) 12-22.
[22] E. Arisholm, L. C.Briand, S. E. Hove, Y. Labiche, The impact of UML documentation on software maintenance: An experimental evaluation, IEEE Transactions on Software Engineering, 32(6) (2006) 365-381.

[23] L. Argote, P. Ingram, J. M. Levine, R. L. Moreland, Knowledge transfer in organizations, Organizational Behavior and Human Decision Processes, 82(1) (2000) 1-8.
[24] M. Aldanondo, S. Rouge, M. Ve, Expert configurator for concurrent engineering: Came´ le´ on software and model, Journal of Intelligent Manufacturing, 11(2) (2000)127-134.
[25] P. Chao, T. Te Chen, Analysis of assembly through product configuration, Computers in Industry, 44(2) (2001) 189-203.
[26] D. Magro, P. Torasso, Decomposition strategies for configuration problems, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 17(01) (2003) 51-73.
[27] H. Tseng, C. Chang, S. Chang, Applying case-based reasoning for product configuration in a mass customization environment, Expert Systems with Applications, 29(4) (2005) 913-925.
[28] T. Guðlaugsson, P. Ravn, N. Mortensen, R. Sarban, Front-end conceptual platform modelling, Concurrent Engineering, 22(4) (2014) 267-276.
[29] M. Mekhilef, J. Bourey, M. Bigand, An UML Modelling of an Architecture for Knowledge Documentation, in: International Conference on Engineering Design (ICED), Stockholm, 2003.
[30] A. Felfernig, D. Jannach, M. Zanker, Contextual diagrams as structuring mechanisms for designing configuration knowledge bases in UML, in: International Conference on the Unified Modeling Language, Springer, 2000, pp. 240-254.
[31] L. Hvam, A procedure for the application of product modelling, International Journal of Production Research, 39(5) (2001) 873-885.
[32] K. Beck, W. Cunningham, A laboratory for teaching object-oriented thinking, ACM Sigplan Notices, 24(10) (1989) 1-6.
[33] L. Hvam, J. Riis, B. Hansen, CRC cards for product modelling. Computers in Industry, 50(1) (2003) 57-70.
[34] B. Renzl, Trust in management and knowledge sharing: The mediating effects of fear and knowledge documentation, Omega, 36(2) (2008) 206-220.
[35] Rask, I. (1998). Rule-Based Product Development. Molndal: Industrial Research and Development Corporation.
[36] S. Sunnersjo, M. Cederfeldt, F. Elgh, I. Rask, A transparent design system for iterative product development, Journal of Computing and Information Science in Engineering, 6(3) (2006) 300-307.
[37] d. Kuhn, Selecting and effectively using a computer aided software engineering tool, in: Annual Westinghouse Computer Symposium. Pittsburgh, PA, USA, 1989.
[38] R.J. Kusters, G. Wijers, On the practical use of CASE-tools: Results of a survey, in: IEEE Proceeding of the Sixth International Workshop on Computer-Aided Software Engineering, 1993.
[39] F. Elgh, Decision support in the quotation process of engineered-to-order products, Advanced Engineering Informatics, 26(2012) (2011) 66-79.
[40] C.J. Paredis, T. Johnson, Using OMG's SysML to support simulation, in: Simulation Conference, IEEE, 2011, pp. 2350-2352.
[41] R. Karban, T. Weilkiens, R. Hauber, M. Zamparelli, M. Diekmann, A. Hein, MBSE Initiative–SE2 Challenge Team–Cookbook for MBSE with SysML, SE2 Challenge Team, 2011.
[42] S.C. Spangelo, D. Kaslow, C. Delp, B. Cole, L. Anderson, E. Fosse, B.S. Gilbert, L. Hartman, T. Kahn, J. Cutler, Applying model based systems engineering (mbse) to a standard cubesat, in: Aerospace Conference, IEEE, 2012, pp. 1-20.
[43] C.J. Paredis, Y. Bernard, R.M. Burkhart, H.P. de Koning, S. Friedenthal, P. Fritzson, An overview of the SysML-modelica transformation specification, in: 2010 INCOSE international symposium, 2010.
[44] D. Stelzer, W. Mellis, Success factors of organizational change in software process improvement, Software Process Improvement and Practice, 4(4) (1998) 227-250.
[45] A. Cockburn, Agile software development joins the 'would-be' crowd, Cutter IT Journal, 15(1) (2002) 6-12.
[46] B. Selic, Agile documentation, anyone?, IEEE Software, 26(6) (2009) 11-12.

[47] S. Ambler, Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process. New York, Wiley Computer Publishing (John Wiley & Sons, Inc.), 2002.
[48] L. Briand, Software documentation: How much is enough?, in: IEEE European Conference on Software Maintenance and Reengineering, 2003.
[49] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration: From Research to Business Cases. Morgan Kaufmann, 2014.
[50] V. Myllärniemi, J. Tiihonen, M. Raatikainen, A. Felfernig, Using answer set programming for feature model representation and configuration, in: International Configuration Workshop, CEUR Workshop Proceedings. Novi Sad, Serbia, 2014.
[51] L. Hvam, S. P. Christensen, K. L. Jensen, J. Riis, Development and maintenance of product configuration systems: Requirements for a documentation tool, International Journal of Industrial Engineering: Theory Applications and Practice, 12(1) (2005) 79-88.
[52] B. Pentland, Information systems and organizational learning: The social epistemology of organizational knowledge systems, Accounting, Management and Information Technology, 5(1) (1995) 1-21.
[53] T. Hansen, Supporting the CPM Procedure. Lyngby, Denmark, Technical University of Denmark, 2010.
[54] A.B. Brown, A. Keller, J.L. Hellerstein, A model of configuration complexity and its application to a change management system, in: IFIP/IEEE International Symposium on Integrated Network Management, IEEE, 2005, pp. 631-644.
[55] P. Comptont, R. Jansen, A philosophical basis for knowledge acquisition, Knowledge Acquisition, 2 (1990) 241-257.